Introduction to SheetSmart

---Revolutionizing Teacher's Spreadsheet Management

In the ever-evolving landscape of education, effective tools and resources play a pivotal role in simplifying the complex tasks teachers face daily. Recognizing this need for efficiency and precision, I present SheetSmart, a groundbreaking project designed exclusively for educators.

SheetSmart is a cutting-edge platform meticulously crafted to empower teachers with the ability to effortlessly manage and create spreadsheets tailored to their unique needs. It not only simplifies the process but also offers an unprecedented level of control and accuracy, ensuring that every piece of data aligns perfectly with your expectations.

Time-Saving Efficiency:

SheetSmart is a time-saving marvel, allowing educators to streamline their administrative tasks and allocate more precious time to their core teaching responsibilities.

Customized Solutions:

With its adaptable features, SheetSmart provides educators with the flexibility to customize spreadsheets, making them the ideal tools for specific classroom needs, whether it's attendance tracking, grading, or data analysis.

Data Precision:

Precision is paramount in education, and SheetSmart ensures that every data entry, calculation, and graph is executed with utmost accuracy, minimizing room for error.

User-Friendly Interface:

The platform boasts an intuitive and user-friendly interface, making it accessible and manageable for educators of all technological backgrounds.

Seamless Integration:

SheetSmart seamlessly integrates with existing educational software and systems, allowing for smooth data sharing and compatibility, reducing workflow disruptions.

Continuous Improvement:

As education evolves, so does SheetSmart . Regular updates and improvements ensure that educators stay ahead in the ever changing landscape of teaching tools.

Key Features of SheetSmart

1. Streamlined Spreadsheet Creation:

With SheetSmart, teachers can create spreadsheets with ease, eliminating the need for manual data entry. The system is designed to be intuitive, making even the most complex tasks manageable.

2. Data Accuracy:

SheetSmart boasts an advanced accuracy feature that verifies data changes within the program and automatically commits them to the associated spreadsheets. This ensures that your data is always up-to-date and error-free.

3. Bi-Directional Compatibility:

Seamlessly interchange data between SheetSmart and spreadsheets. Make changes in the program, and they reflect in your sheets instantly. Edit your spreadsheets, and SheetSmart absorbs the updates, facilitating effortless sharing and collaboration.

4.Versatile Sharing:

SheetSmart allows you to share your spreadsheets across various networks and media, enhancing collaboration and making information more accessible than ever before.

5.User Testimonials:

The success of SheetSmart is not just our claim; it's backed by glowing testimonials from educators who have experienced its transformative power. They speak of increased productivity, reduced workload, and enhanced precision in their daily tasks, attesting to the platform's effectiveness.

6.Educational Institution Adoption:

Numerous educational institutions have integrated SheetSmart into their teaching practices, demonstrating its wide acceptance and impact in the education sector. These institutions cite improved data management, collaborative advantages, and overall efficiency as key reasons for adopting SheetSmart as an essential tool for educators.

SheetSmart is poised to revolutionize the way teachers handle their data, bringing efficiency, accuracy, and convenience to the forefront.

Proposed System SheetSmart

Empowering Teachers with Effortless Spreadsheet Management In today's educational landscape, educators are entrusted with the crucial task of shaping the future. However, the sheer volume of administrative work, data management, and record-keeping can often be overwhelming. The adage "to err is human" no longer applies in this highly competitive and unforgiving environment, and teachers need a solution that allows them to excel without operational hindrances.

Enter SheetSmart, a dedicated project meticulously designed to cater to the specific needs of teachers. Its primary objective is to simplify the process of creating, managing, and editing spreadsheets, enhancing teachers' daily workflow and, in turn, the quality of education they deliver.

Key Highlights of SheetSmart

Effortless Spreadsheet Creation:

SheetSmart offers teachers an intuitive platform for creating spreadsheets without the need for complex manual data entry. This feature is engineered to make even the most intricate tasks straightforward.

Data Accuracy at Your Fingertips:

With SheetSmart, data precision is paramount. Any changes made within the program are seamlessly committed to the associated spreadsheets, ensuring that your data is always error-free and up to date.

Bidirectional Compatibility:

Seamlessly transfer data between SheetSmart and spreadsheets. Whether you edit within the program or directly in the spreadsheet, SheetSmart maintains synchronization, facilitating effortless sharing and collaboration.

Effortless Sharing Across Networks:

SheetSmart ensures the ease of sharing your data across various media and networks, enabling better collaboration and access.

SheetSmart has been meticulously crafted to serve teachers, ensuring that their valuable time and resources are optimally utilized. By adopting this innovative tool, teachers experience a significant reduction in time spent on administrative tasks, freeing them to focus on what truly matters: educating and nurturing the next generation.

In the era of education powered by technology and automation, SheetSmart stands as a beacon of efficiency and professionalism. By integrating this powerful tool into their daily routine, teachers not only enhance their productivity but also present a modern and tech-savvy image to their students and peers.

In summary, SheetSmart is exclusively tailored to teachers, providing them with an efficient, user-friendly, and teacher-centric approach to spreadsheet management. This project is a testament to my commitment to enhancing the teaching experience, minimizing administrative overhead, and empowering educators in an ever-competitive educational landscape.

OBJECTIVES OF THE PROJECT TO STUDENTS

The SheetSmart project is more than just a software development venture; it's a platform to empower students to apply their programming knowledge in practical, real-world scenarios. By engaging in this project, students will have the opportunity to grasp how programming skills translate into the creation of efficient software solutions. My objectives include:

Practical Application:

To enable students to write programs using contemporary software tools, applying their coding skills to real-world problems.

Effective Software Development:

To help students effectively implement objectoriented programming principles when developing small to medium-sized projects, instilling good coding practices.

Problem-Solving Proficiency:

To equip students with the ability to write efficient procedural code for tackling small to medium-sized issues, honing their problem-solving skills.

Comprehensive Knowledge:

To encourage students to broaden their knowledge in computer science, covering areas of systems, theory, and software development, fostering well-rounded expertise.

Research and Presentation Skills:

To develop students' abilities to conduct research or applied Computer Science projects, emphasizing scholarly writing and presentation skills, essential in the world of academia and professional development.

SheetSmart is not just a software tool; it's a hands-on educational experience designed to cultivate a deep understanding of programming's practical applications, coding principles, and problem-solving capabilities in the digital age.

SheetSmart goes beyond being a mere software tool; it's a gateway to a realm of educational exploration. It serves as a medium for students to delve into the practical applications of programming, allowing them to grasp the intricacies of coding principles, problem-solving, and data management. In an era defined by digital innovation, SheetSmart is the bridge that connects classroom knowledge to real-world proficiency, empowering students to navigate the challenges of the modern age with confidence and competence.

OBJECTIVES OF THE SHEETSMART

SheetSmart is not just a project; it's a key to unlocking a world of convenience and efficiency. Whether you're a student eager to streamline your coursework or a budding tech enthusiast, SheetSmart offers a user-friendly platform to master spreadsheet management. With this tool at your disposal, you'll be able to create, edit, and share data effortlessly. So, embark on this journey with us and discover the transformative potential of SheetSmart

SheetSmart isn't just about numbers and cells; it's a game-changer for educators. Teachers, imagine a world where your administrative workload diminishes, allowing you to focus more on teaching and nurturing your students. SheetSmart simplifies data management, ensuring accuracy and facilitating data sharing across networks. It's a powerful tool designed exclusively for you, equipping you with the means to enhance your productivity and ultimately provide a better educational experience for your students.

In this digital age, knowledge is the currency that fuels productivity. As an individual taking on the SheetSmart project alone, you're on a journey to gain expertise in software development and data management. By mastering SheetSmart, you're equipping yourself with a valuable skill set, staying updated with the latest technology, and enhancing your productivity. This project is your personal gateway to becoming a tech-savvy educator and a more effective contributor to the world of education.

SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)

The Software Development Life Cycle (SDLC) is a structured approach to managing and executing complex projects. It breaks down intricate endeavors into more manageable segments or phases, ensuring that each phase is successfully completed before resources are allocated to the subsequent stages. While the standard phases in software development encompass initiation, planning, design, development, testing, implementation, and maintenance, it's important to note that the division of phases may vary depending on the specific organization and project. For instance, initial project activities could be categorized as request, requirements-definition, and planning phases, or even initiation, concept development, and planning phases. Crucially, the involvement of end users throughout the process is vital to review the output of each phase, guaranteeing that the final system aligns with the intended functionality.

PHASES OF SYSTEM DEVELOPMENT LIFE CYCLE INITIATION PHASE

Opportunity Identification:

The Initiation Phase begins when a business sponsor identifies a need or an opportunity.

Purpose of Initiation:

The purpose of this phase is to identify and validate opportunities for improving business accomplishments or addressing deficiencies related to a business need.

Assumptions and Constraints:

Significant assumptions and constraints on potential solutions are identified and considered during this phase.

Exploring Alternatives:

The phase recommends exploring alternative concepts and methods to satisfy the need, including assessing whether a change in the business process alone could offer a solution without the need for technology.

Executive Sponsorship:

Assurance of executive business and technical sponsorship is a critical component of the Initiation Phase.

Project Manager Designation:

The sponsor designates a Project Manager, and the business need is documented in a Concept Proposal.

Concept Proposal Content:

The Concept Proposal includes information about the business process, its relationship to the organization, infrastructure, and the strategic plan.

Project Management Charter:

A successful Concept Proposal results in the creation of a Project Management Charter that outlines the authority of the Project Manager to initiate the project.

Business Case Presentation:

The initiation phase begins when an opportunity to add, improve, or correct a system is identified and formally requested through the presentation of a business case.

Business Case Elements:

The business case should describe the proposal's purpose, identify expected benefits, and explain how the proposed system supports one of the organization's business strategies.

Alternative Solutions:

The business case should also identify alternative solutions and detail informational, functional, and network requirements as comprehensively as possible.

SYSTEM CONCEPT DEVELOPMENT PHASE

Project Conceptualization:

The System Concept Development Phase marks the beginning of a project where the initial idea or concept is conceptualized.

Idea Refinement:

During this phase, the project concept is refined and detailed, with a focus on clarifying the objectives and requirements.

Feasibility Assessment:

Feasibility studies may be conducted to assess the project's viability, including technical, operational, and financial feasibility.

Requirements Gathering:

Requirements for the project are identified and documented, specifying what the system should achieve and how it should function.

Stakeholder Involvement:

Collaboration with stakeholders, including end users, is crucial to gather insights, feedback, and align the project concept with their needs.

High-Level Planning:

Initial project planning efforts begin, encompassing highlevel project planning, resource allocation, and broad scheduling.

Risk Assessment:

Potential risks and challenges are assessed to identify potential obstacles that may arise during the development phase.

Proof of Concept:

In complex projects, a proof of concept or prototype may be developed to demonstrate the feasibility of the proposed system.

Concept Documentation:

The refined project concept is documented in detail, creating a comprehensive project concept document that serves as a guide for further development.

Concept Validation:

The project concept is reviewed and validated to ensure it aligns with the organization's strategic goals and that it's worth pursuing.

Decision Point:

At the end of this phase, a decision is made regarding whether to proceed with the project based on the concept's viability, alignment with objectives, and available resources.

PLANNING PHASE

The Planning Phase is a critical stage in any project, where meticulous groundwork is laid for success. During this phase, project objectives are defined, resources are allocated, and detailed planning takes place. It is in this phase that the project's roadmap is carefully charted, from setting clear milestones to assigning tasks and budgeting. Effective planning is the linchpin of successful project management, as it ensures that every aspect of the project aligns with the overall goals and expectations. With careful planning, projects are more likely to stay on track, meet deadlines, and achieve their intended outcomes.

In the realm of systems engineering management, meticulous planning sets the stage for success. It's in this critical phase that project objectives are clearly defined, resources are allocated, and detailed planning takes center stage. As we transition into the Requirements Analysis Phase, the project delves into the crucial task of gathering, documenting, and comprehensively understanding the specific needs and constraints that will shape its design and execution. This phase serves as the bridge between conceptualization and actual implementation, where the project's scope and parameters are refined to ensure that it aligns perfectly with its intended objectives. With a strong foundation in systems engineering management and thorough requirements analysis, projects are poised for effective execution and successful outcomes.

DEVELOPMENT PHASE

As the project advances into the Development Phase, the meticulously crafted design specifications come to life in the form of functional software.

Developers, armed with the design blueprints and coding principles, embark on scripting the software components. This phase employs a combination of topdown and bottom-up approaches, ensuring the seamless integration of major program components, interfaces, and subsystems. Modern development techniques often utilize prototyping tools to create mock-up versions of the software's key elements, allowing for iterative refinement in collaboration with end users, designers, developers, database managers, and network administrators. The goal is to satisfy the functional requirements outlined in the earlier phases, keeping in mind that issues identified in the design phase can be costly to address in later stages. Risk mitigation takes center stage, encompassing identifying potential risks, conducting security assessments, devising data migration plans, defining the operating environment, and allocating processes to resources. Detailed logic specifications for each software module are prepared. This phase culminates in the creation of a draft System Design Document, capturing the preliminary design. All user inputs and approvals are documented, leading to the final System Design Document, which undergoes rigorous technical and functional reviews to ensure alignment with business requirements. Concurrently, the Agency Project Manager initiates the development of essential project documents, including the Implementation Plan, Operations and Maintenance Manual, and the Training Plan, ensuring a comprehensive and efficient software development process.

INTEGRATION AND TEST PHASE

In the Integration and Test Phase, the project reaches a crucial stage where design specifications and development efforts converge, resulting in the creation of executable programs. This phase emphasizes robust development standards that entail comprehensive discussions among programmers and project stakeholders to ensure a clear understanding of program designs and functional requirements before coding begins.

Development Techniques

Developers employ a range of techniques to craft computer programs, with large transaction-oriented programs, often associated with financial institutions, traditionally relying on procedural programming methods. Procedural programming involves the systematic scripting of logical instructions, which are then woven together to form a cohesive program.

Key Activities

The Development phase encompasses several key activities, including:
- Translating detailed requirements and designs into system components.
- Testing individual elements (units) for usability.
- Preparing for the integration and testing of the IT system.

INTEGRATION AND TEST PHASE

The Integration and Test Phase is a critical juncture in the project's development journey, where the meticulously crafted design specifications and development efforts culminate in the assembly of executable programs.

VALIDATION AND ACCREDITATION

Subsystem integration, system, security, and user acceptance testing take center stage during this phase. Users, in collaboration with quality assurance stakeholders, validate that the developed or modified system aligns with the functional requirements outlined in the functional requirements document. Security assessments, including certification and accreditation, are conducted by OIT Security staff before installation/implementation.

Multiple Levels of Testing

Multiple levels of testing are executed, including:

- Testing at the development facility with contractor and end-user support.
- Testing as a deployed system with end users and contract personnel.
- Operational testing by end users working autonomously.

The entire testing process maintains a strict focus on tracing requirements, culminating in a final Independent Verification & Validation evaluation and comprehensive documentation review before system acceptance.

IMPLEMENTATION PHASE

The Implementation Phase is initiated post-testing, following user acceptance.

Installation and Performance

During this phase, the system is installed to facilitate the intended business functions. Performance is compared to objectives set during the planning phase.

Key Activities

- User notifications and training.
- Hardware and software installations on production computers.
- Seamless system integration into daily work processes.

This phase endures until the system is fully operational and aligns with the defined user requirements.

OPERATIONS AND MAINTENANCE PHASE

As operations commence, the Operations and Maintenance Phase takes center stage.

Ongoing Monitoring

The system's ongoing operation is closely monitored to ensure performance in line with user requirements. Any necessary system modifications are diligently incorporated.

Long-Term Operation

This phase continues as long as the system remains adaptable to the organization's evolving needs.

Periodic Assessments

Periodic assessments are conducted to verify the continued satisfaction of functional requirements.

Modernization and Retirement Considerations

When the system's capacity to respond to the organization's needs wanes, modernization, replacement, or retirement considerations are deliberated.

Primary Goals

The primary goals of this phase are to:

- Operate, maintain, and enhance the system.
- Certify its capability to process sensitive information.
- Periodically assess its functionality to ensure ongoing satisfaction of functional requirements.

SOURCE CODE

SheetSmart.py

```
import csv
def write_csv():
    file=open('sheetsmart.csv','a')
    wo=csv.writer(file)
    ans='y'
    while ans.lower()=='y':
        l=[]
        rno=int(input('Enter roll number: '))
        name=input('Enter name: ')
        eng=float(input('Enter English mark: '))
        maths=float(input('Enter Maths mark: '))
        phy=float(input('Enter Physics mark: '))
        che=float(input('Enter Chemistry mark: '))
        cs=float(input('Enter CS mark: '))
        total=eng+maths+phy+che+cs
        average=total/5
        if average >= 90:
            grade = 'A+'
        elif average >= 80:
            grade = 'A'
        elif average >= 70:
            grade = 'B'
        elif average >= 60:
            grade = 'C'
        else:
            grade = 'D'
        l=[rno,name,eng,maths,phy,che,cs,grade,average]
        wo.writerow(l)
        file.flush()
        print('Data of the',l[1],'has been written to the storage Sucessfully!!!')
        ans=input('Do you Want to Continue (y/n): ')
    file.close()
def change_csv_cs():
    rno=int(input('Enter the roll number: '))
    cs = float(input('Enter new CS mark: '))
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    cs_found = False
    for row in data:
        if str(rno)==row[0]:
                row[6] = str(cs)
                total = float(row[2]) + float(row[3]) + float(row[4]) + float(row[5]) + float(row[6])
                row[8] = str(total / 5)
```

```
            average=total/5
            if average >= 90:
                grade = 'A+'
            elif average >= 80:
                grade = 'A'
            elif average >= 70:
                grade = 'B'
            elif average >= 60:
                grade = 'C'
            else:
                grade = 'D'
            row[7] = grade
            cs_found = True
    if cs_found:
        with open('sheetsmart.csv', 'w', newline='\n') as file:
            wo = csv.writer(file)
            wo.writerows(data)
        print('CS mark change has been written to the storage successfully!')
    else:
        print('rno not found in the storage.')
def change_csv_eng():
    rno=int(input('Enter the roll number: '))
    eng = float(input('Enter new English mark: '))
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    eng_found = False
    for row in data:
        if str(rno)==row[0]:
            row[3] = str(eng)
            total = float(row[2]) + float(row[3]) + float(row[4]) + float(row[5]) + float(row[6])
            row[8] = str(total / 5)
            average=total/5
            if average >= 90:
                grade = 'A+'
            elif average >= 80:
                grade = 'A'
            elif average >= 70:
                grade = 'B'
            elif average >= 60:
                grade = 'C'
            else:
                grade = 'D'
            row[7] = grade
            eng_found = True
    if eng_found:
        with open('sheetsmart.csv', 'w', newline='\n') as file:
            wo = csv.writer(file)
            wo.writerows(data)
        print('English mark change has been written to the storage successfully!')
```

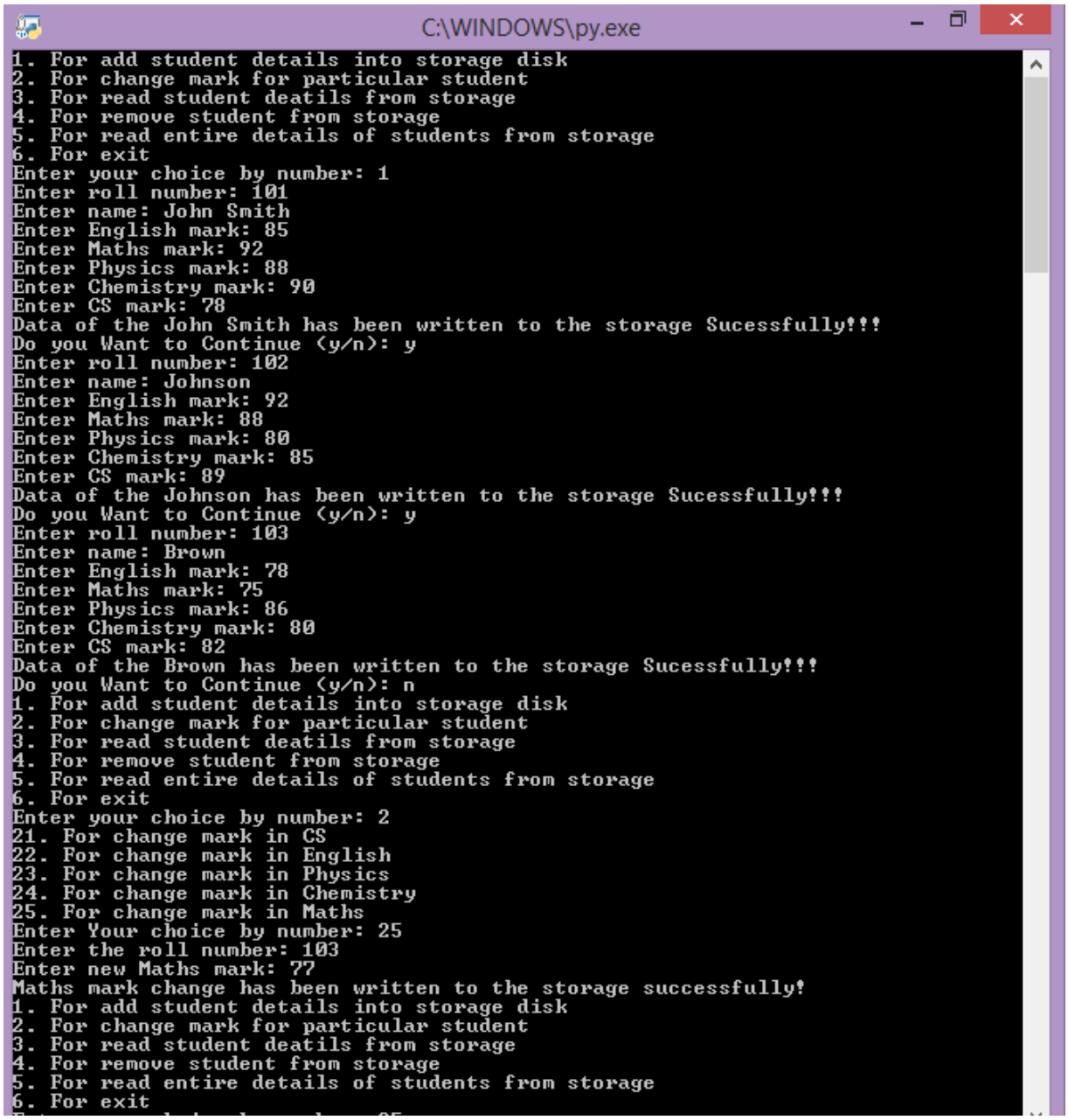
```
    else:
        print('rno not found in the storage.')
def change_csv_maths():
    rno=int(input('Enter the roll number: '))
    maths = float(input('Enter new Maths mark: '))
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    maths_found = False
    for row in data:
        if str(rno)==row[0]:
                row[4] = str(maths)
                total = float(row[2]) + float(row[3]) + float(row[4]) + float(row[5]) + float(row[6])
                row[8] = str(total / 5)
                average=total/5
                if average >= 90:
                    grade = 'A+'
                elif average >= 80:
                    grade = 'A'
                elif average >= 70:
                    grade = 'B'
                elif average >= 60:
                    grade = 'C'
                else:
                    grade = 'D'
                row[7] = grade
                eng_found = True
    if eng_found:
        with open('sheetsmart.csv', 'w', newline='\n') as file:
            wo = csv.writer(file)
            wo.writerows(data)
        print('Maths mark change has been written to the storage successfully!')
    else:
        print('rno not found in the storage.')
def change_csv_phy():
    rno=int(input('Enter the roll number: '))
    phy = float(input('Enter new Physics mark: '))
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    phy_found = False
    for row in data:
        if str(rno)==row[0]:
                row[5] = str(phy)
                total = float(row[2]) + float(row[3]) + float(row[4]) + float(row[5]) + float(row[6])
                row[8] = str(total / 5)
                average=total/5
                if average >= 90:
                    grade = 'A+'
                elif average >= 80:
```

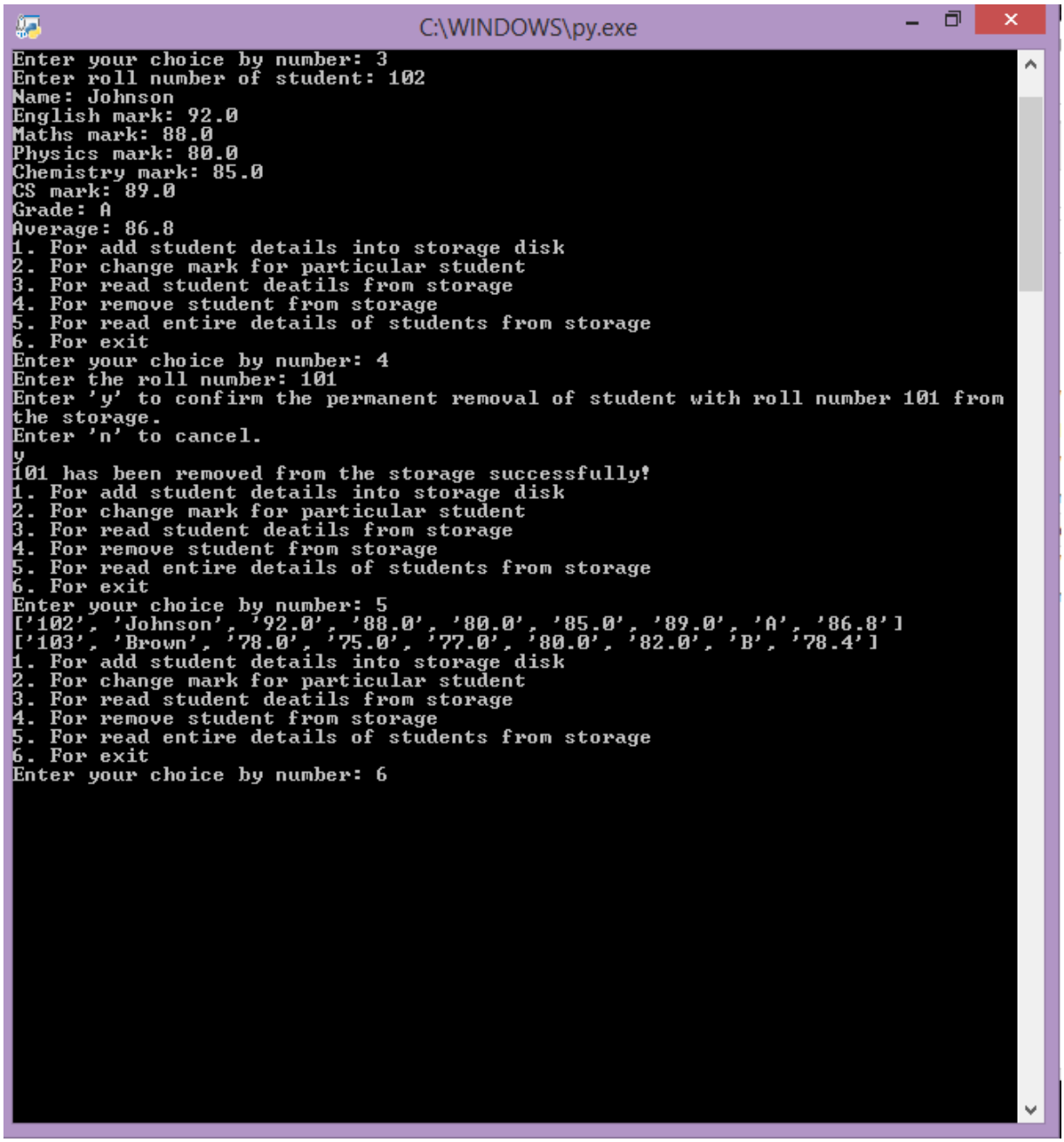
```
                grade = 'A'
            elif average >= 70:
                grade = 'B'
            elif average >= 60:
                grade = 'C'
            else:
                grade = 'D'
            row[7] = grade
            eng_found = True
    if eng_found:
        with open('sheetsmart.csv', 'w', newline='\n') as file:
            wo = csv.writer(file)
            wo.writerows(data)
        print('Physics mark change has been written to the storage successfully!')
    else:
        print('rno not found in the storage.')
def change_csv_che():
    rno=int(input('Enter the roll number: '))
    che = float(input('Enter new Chemistry mark: '))
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    che_found = False
    for row in data:
        if str(rno)==row[0]:
            row[5] = str(che)
            total = float(row[2]) + float(row[3]) + float(row[4]) + float(row[5]) + float(row[6])
            row[8] = str(total / 5)
            average=total/5
            if average >= 90:
                grade = 'A+'
            elif average >= 80:
                grade = 'A'
            elif average >= 70:
                grade = 'B'
            elif average >= 60:
                grade = 'C'
            else:
                grade = 'D'
            row[7] = grade
            che_found = True
    if che_found:
        with open('sheetsmart.csv', 'w', newline='\n') as file:
            wo = csv.writer(file)
            wo.writerows(data)
        print('Chemistry mark change has been written to the storage successfully!')
    else:
        print('rno not found in the storage.')
def read_details_csv():
    rno=int(input('Enter roll number of student: '))
```

```
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    details_found = False
    for student_details in data:
        if str(rno)==student_details[0]:
            print('Name:',student_details[1])
            print('English mark:',student_details[2])
            print('Maths mark:',student_details[3])
            print('Physics mark:',student_details[4])
            print('Chemistry mark:',student_details[5])
            print('CS mark:',student_details[6])
            print('Grade:',student_details[7])
            print('Average:',student_details[8])
            details_found = True
    if details_found:
        pass
    else:
        print('roll numer is not there in the storage.')
def remove_csv():
    rno=int(input('Enter the roll number: '))
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    remove_found = False
    for row in data:
        if str(rno)==row[0]:
                print("'Enter 'y' to confirm the permanent removal of student with roll number'", rno,
"'from the storage.\nEnter 'n' to cancel.'")
                conform=input()
                if conform.lower() == 'y':
                    data.remove(row)
                    remove_found = True
                else:
                    pass
    if remove_found:
        with open('sheetsmart.csv', 'w', newline='\n') as file:
            wo = csv.writer(file)
            wo.writerows(data)
        print(rno,'has been removed from the storage successfully!')
    else:
        print('roll number not found in the storage.')
def read_csv():
    with open('sheetsmart.csv', 'r', newline='\n') as file:
        ro = csv.reader(file)
        data = list(ro)
    for row in data:
        print(row)
while True:
    print('1. For add student details into storage disk')
```

```
print('2. For change mark for particular student')
print('3. For read student deatils from storage')
print('4. For remove student from storage')
print('5. For read entire details of students from storage')
print('6. For exit')
ch=int(input('Enter your choice by number: '))
if ch == 1:
    write_csv()
elif ch == 2:
    print('21. For change mark in CS')
    print('22. For change mark in English')
    print('23. For change mark in Physics')
    print('24. For change mark in Chemistry')
    print('25. For change mark in Maths')
    ch=int(input('Enter Your choice by number: '))
    if ch == 21:
        change_csv_cs()
    elif ch == 22:
        change_csv_eng()
    elif ch == 23:
        change_csv_phy()
    elif ch == 24:
        change_csv_che()
    elif ch == 25:
        change_csv_maths()
    else:
        print('You Enter a Wrong number!!!')
elif ch == 3:
    read_details_csv()
elif ch == 4:
    remove_csv()
elif ch == 5:
    read_csv()
else:
    break
```

OUTPUT


C:\WINDOWS\py.exe
1. For add student details into storage disk
2. For change mark for particular student
3. For read student deatils from storage
4. For remove student from storage
5. For read entire details of students from storage
6. For exit
Enter your choice by number: 1
Enter roll number: 101
Enter name: John Smith
Enter English mark: 85
Enter Maths mark: 92
Enter Physics mark: 88
Enter Chemistry mark: 90
Enter CS mark: 78
Data of the John Smith has been written to the storage Sucessfully!!!
Do you Want to Continue (y/n): y
Enter roll number: 102
Enter name: Johnson
Enter English mark: 92
Enter Maths mark: 88
Enter Physics mark: 80
Enter Chemistry mark: 85
Enter CS mark: 89
Data of the Johnson has been written to the storage Sucessfully!!!
Do you Want to Continue (y/n): y
Enter roll number: 103
Enter name: Brown
Enter English mark: 78
Enter Maths mark: 75
Enter Physics mark: 86
Enter Chemistry mark: 80
Enter CS mark: 82
Data of the Brown has been written to the storage Sucessfully!!!
Do you Want to Continue (y/n): n
1. For add student details into storage disk
2. For change mark for particular student
3. For read student deatils from storage
4. For remove student from storage
5. For read entire details of students from storage
6. For exit
Enter your choice by number: 2
21. For change mark in CS
22. For change mark in English
23. For change mark in Physics
24. For change mark in Chemistry
25. For change mark in Maths
Enter Your choice by number: 25
Enter the roll number: 103
Enter new Maths mark: 77
Maths mark change has been written to the storage successfully!
1. For add student details into storage disk
2. For change mark for particular student
3. For read student deatils from storage
4. For remove student from storage
5. For read entire details of students from storage
6. For exit


C:\WINDOWS\py.exe
Enter your choice by number: 3
Enter roll number of student: 102
Name: Johnson
English mark: 92.0
Maths mark: 88.0
Physics mark: 80.0
Chemistry mark: 85.0
CS mark: 89.0
Grade: A
Average: 86.8
1. For add student details into storage disk
2. For change mark for particular student
3. For read student deatils from storage
4. For remove student from storage
5. For read entire details of students from storage
6. For exit
Enter your choice by number: 4
Enter the roll number: 101
Enter 'y' to confirm the permanent removal of student with roll number 101 from the storage.
Enter 'n' to cancel.
y
101 has been removed from the storage successfully!
1. For add student details into storage disk
2. For change mark for particular student
3. For read student deatils from storage
4. For remove student from storage
5. For read entire details of students from storage
6. For exit
Enter your choice by number: 5
['102', 'Johnson', '92.0', '88.0', '80.0', '85.0', '89.0', 'A', '86.8']
['103', 'Brown', '78.0', '75.0', '77.0', '80.0', '82.0', 'B', '78.4']
1. For add student details into storage disk
2. For change mark for particular student
3. For read student deatils from storage
4. For remove student from storage
5. For read entire details of students from storage
6. For exit
Enter your choice by number: 6

TESTING: Uncovering Software Quality

Software Testing is a rigorous, empirical investigation conducted to furnish stakeholders with invaluable insights into the quality of the product or service being tested. This assessment is carried out within the context in which the software is intended to function. Furthermore, it provides a detached and objective perspective on the software, aiding businesses in comprehending and mitigating potential risks during implementation.

Testing encompasses a variety of techniques, including the execution of software or applications with the specific goal of identifying and rectifying software defects. It also entails validating and verifying that a software program, application, or product aligns with both the business and technical requirements that guided its conception and development. Ultimately, the goal is to ensure that it operates as intended and is deployable with consistent performance. Software Testing is a flexible process that can be executed at different stages of development, with the most significant effort typically invested after requirements have been defined and the coding process is completed.

TESTING METHODS:

Unraveling Software Testing Approaches Software testing methods are traditionally classified into two main categories: black box testing and white box testing. These categories describe the perspective taken by a test engineer when designing test cases.

BLACK BOX TESTING: The Unseen Tester

Black box testing treats the software as an enigmatic "black box" without delving into its internal implementation. Notable black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing, and specification-based testing.

SPECIFICATION-BASED TESTING: Aligning with Requirements

Specification-based testing primarily focuses on validating software functionality according to the specified requirements. In this approach, testers input data into the software and assess the output against expected results, adhering to the test cases provided. While specification-based testing is essential, it alone may not be sufficient to mitigate all potential risks.

ADVANTAGES AND DISADVANTAGES: Navigating the Complexities

In black box testing, testers remain impartial to the code, approaching it with the expectation of uncovering bugs. However, this detachment can lead to situations where extensive test cases are written when a single test case could suffice, or where certain parts of the software's backend remain untested.

WHITE BOX TESTING: Unveiling the Code

In contrast, white box testing provides testers access to internal data structures, algorithms, and the underlying code. This approach allows testers to probe deeper into the software, ensuring comprehensive coverage.

TYPES OF WHITE BOX TESTING: A Comprehensive View

Various forms of white box testing exist, including API testing, code coverage, fault injection methods, mutation testing methods, and static testing, which encompasses all forms of static testing.

CODE COMPLETENESS EVALUATION: Ensuring Thorough Testing

White box testing methods can also be employed to assess the completeness of a test suite originally created using black box testing methods. This serves as a valuable check to ensure that critical function points have been rigorously tested. Two common metrics for code coverage include function coverage, which reports on executed functions, and statement coverage, which measures the number of lines executed during testing.

By adopting a mix of black box and white box testing, software quality can be effectively evaluated and improved, ultimately contributing to a robust and reliable product or service.

SYSTEM REQUIREMENTS FOR SHEETSMART APPLICATION

I. Operating System: Windows 7 or later, any type of Linux distribution, macOS, Chrome OS and other widely used operating systems.

II. Processor: Pentium (Any) or AMD Athlon (3800+ to 4200+ Dual-Core)

III. Motherboard:

- For Pentium: 845 or 915, 995
- For AMD Athlon: MSI K9MM-V with VIA K8M800+8237R Plus Chipset

IV. RAM: 128 MB or higher

V. Hard Disk: 512 MB is minimum

VI. CD/DVD R/W Multi-Drive Combo: (Optional, if backup functionality is required)

VII. Floppy Drive 1.44 MB: (Optional, if backup functionality is required)

VIII. Monitor: 14.1-inch or 15 to 17-inch display

IX. Keyboard and Mouse: Standard input devices

X. Printer: (Optional, if hard copy printing is required)

These system requirements ensure that the SheetSmart application is compatible with a wide range of operating systems used around the world, allowing users to choose their preferred platform for using SheetSmart.

SOFTWARE REQUIREMENTS

1. Python

INSTALLATION PROCEDURE

1. Install Python from python.org
2. Run sheetsmart.py file from your device
3. Experience the power of efficiency and innovation with SheetSmart.

PYTHON

Python is a versatile and widely-used programming language known for its simplicity and readability. It was created by Guido van Rossum, a Dutch programmer, and was first released in 1991. Python's design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than languages like C++ or Java. Python is widely used in web development, scientific computing, data analysis, artificial intelligence, and more.

Introduction to Python and Guido van Rossum

Python, a high-level, interpreted programming language renowned for its emphasis on code readability and ease of use, was created by Guido van Rossum, a Dutch computer scientist born in 1956. Guido developed Python in the late 1980s and released it to the world in 1991, with the vision of crafting a language that prioritizes simplicity and code readability. Python's design philosophy revolves around meaningful whitespace and provides clear, straightforward constructs, making it an ideal choice for both novice and experienced programmers. Guido's contributions have profoundly influenced the software development community, shaping Python into one of the most popular and widely-used programming languages globally. His work has not only made programming more accessible but has also empowered developers to write elegant and efficient code, leaving a lasting impact across various industries and disciplines.

Guido van Rossum studied at the University of Amsterdam, where he completed his Master's degree in Mathematics and Computer Science. He later earned his Ph.D. in Computer Science from the Centrum Wiskunde & Informatica (CWI) in the Netherlands.


I ♥ pyladies
supporting women
who code

Python: A Language of Versatility

Python is renowned for its versatility, enabling programmers to work on a wide range of projects, from web development and data analysis to artificial intelligence and scientific research. It has gained immense popularity in various domains due to its ease of learning and implementation.

Guido van Rossum: The Father of Python

Guido van Rossum, the creator of Python, is often affectionately referred to as the "Benevolent Dictator For Life" (BDFL) in the Python community. He was born on January 31, 1956, in The Hague, Netherlands. Guido's contribution to the world of programming languages is immeasurable. His journey into the creation of Python was driven by a desire for a language that offered the best of both worlds: simplicity and practicality.

The Genesis of Python

Python's origins can be traced back to the late 1980s when Guido was working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands. He sought to develop a language that was easy to read and write, one that could handle complex tasks while maintaining a clear and straightforward syntax.

In December 1989, Guido began working on Python, taking inspiration from the ABC language developed at CWI. After several iterations and refinements, Python was officially released to the public in February 1991 as Python 0.9.0.

Python's Design Philosophy

One of the standout features of Python is its design philosophy, which is often referred to as the "Zen of Python." This philosophy is encapsulated in a collection of guiding principles that underpin the language's development. Some of the key principles include:

- Readability counts: Python's syntax is designed to be highly readable, emphasizing the importance of clear and comprehensible code.
- Simple is better than complex: Python prioritizes simplicity, encouraging straightforward and efficient solutions.
- There should be one—and preferably only one—obvious way to do it: Python aims to minimize ambiguity and provide a single, obvious way to perform a task.

Python's Ongoing Evolution

Python's development continues to this day, with new versions and updates being regularly released. Guido van Rossum's leadership played a crucial role in shaping the language and its community. His role as the BDFL involved making final decisions on Python's design and direction, creating a sense of coherence within the community.

Guido's Legacy

In July 2018, Guido stepped down from his role as the BDFL, signaling a shift in Python's governance. This transition marked a significant moment in Python's history, but it also highlighted the open and collaborative nature of the Python community, which continues to thrive.

Python's enduring popularity and relevance can be attributed to Guido van Rossum's vision and dedication to creating a programming language that is accessible, powerful, and enjoyable to use. Python's success is a testament to Guido's contribution to the world of software development.

The Python Ecosystem

Python offers a vast ecosystem of libraries and frameworks that extend its capabilities. Topics to explore include the Python Standard Library, NumPy for numerical computing, pandas for data analysis, Django and Flask for web development, and TensorFlow for machine learning.

Python's Role in Web Development

Python is a popular choice for web development, with frameworks like Django and Flask simplifying the process. Explore how Python powers dynamic web applications and content management systems.

Python for Data Science and Analysis

Python's libraries, such as NumPy, pandas, and Matplotlib, make it a powerhouse for data science and analysis. Dive into topics like data manipulation, visualization, and machine learning with Python.

Artificial Intelligence and Machine Learning with Python

Python is widely used in artificial intelligence and machine learning. Discover how Python libraries like TensorFlow, Keras, and PyTorch drive advancements in this field.

Python in Scientific Computing

Python's simplicity and rich ecosystem have made it a go-to choice for scientific computing. Learn about its role in scientific simulations, data analysis, and visualization.

Python in Game Development

Python has made strides in the world of game development. Explore topics like Pygame, a popular library for creating 2D games, and how Python is used for game scripting.

Python for Automation

Python excels in automating tasks. Topics to delve into include scripting, web scraping, and using Python to simplify repetitive tasks.

Cybersecurity and Python

Python is used for ethical hacking, network analysis, and building security tools. Explore its role in cybersecurity and penetration testing.

Python's Role in Blockchain and Cryptocurrency

Python plays a significant role in blockchain development and cryptocurrency projects. Dive into how Python is used to create and manage blockchain applications.

Quantitative Finance and Python

Python is a valuable tool in quantitative finance. Topics to explore include algorithmic trading, risk management, and financial data analysis using Python.

Python for Internet of Things (IoT) Development

Learn how Python is used to develop applications and software for IoT devices, bridging the gap between the physical and digital worlds.

Python's Contribution to Natural Language Processing (NLP)

Python is instrumental in NLP, powering applications like chatbots, sentiment analysis, and language translation. Delve into Python's role in NLP.

Quantum Computing with Python

Python has established itself as a central player in the burgeoning field of quantum computing, offering a versatile and user-friendly environment for quantum programming. Supported by prominent frameworks such as Qiskit, Cirq, and PyQuil, Python facilitates quantum algorithm development, simulations, and analysis. Its rich ecosystem empowers researchers to prototype and experiment with quantum circuits, access quantum algorithm libraries, visualize results, and engage in quantum education and outreach efforts, positioning Python as a pivotal language in advancing quantum computing's frontiers.

Using Python for Big Data Analytics

Python has become a leading language for big data analytics, thanks to libraries like PySpark and Dask, which empower organizations to efficiently process and analyze massive datasets. PySpark, built on top of Apache Spark, provides distributed data processing capabilities, allowing Python developers to harness the power of distributed computing for tasks like data cleansing, transformation, and machine learning on large datasets. Dask, on the other hand, offers parallel and distributed computing functionalities, making it easier to work with large datasets that don't fit into memory. These Python libraries enable data scientists and analysts to leverage the scalability and performance required to extract valuable insights from vast and complex data, ultimately driving data-driven decision-making and innovation.

Bibliography

1. Computer science With Python - Class XI & XII By : SumitaArora
2. chat.openai.com For The Python section of this project
3. peteradamsphoto.com/guido-van-rossum For the photography of Guido Van Rossum